One-jump simulations

Binseginf

Oct 13th, 2016

Check package

```
## Simulation settings
library(binSegInf)
```

```
## Loading required package: Matrix
```

```
## Loading required package: pryr
```

```
## Loading required package: testthat
```

```
n = 12
numsteps = n - 1
nsim = 100
sigma = 0.5
y = rnorm(n,0,sigma)
Bcurr = Zcurr = Scurr = Ecurr =
    Matrix(0, ncol=2^(numsteps+1), nrow = numsteps+1, sparse=TRUE)

verbose=TRUE

    ## Initialize things
    B = Z = rep(NA,length(y))
    Bcurr = Zcurr = Scurr = Ecurr =
        Matrix(0, ncol=2^(numsteps+1), nrow = numsteps+1, sparse=TRUE)
    S = E = A = Tt =
        Tcurr = Acurr =
            lapply(1:length(y),function(i) c())
    Scurr[1,1] = 1
    Ecurr[1,1] = length(y)
    Tcurr[[1]] = c(1,1)
    Acurr[[1]] = c()
    jk  = list()
    zetas = rep(NA,length(y))
    G = matrix(NA, ncol = length(y), nrow = 2*length(y)*numsteps)
    Gn = 0

    ## Main loop
    ## for(mystep in 1:numsteps){
    ##     if(verbose) cat("At step", mystep, " ") ## Get all candidate changepoints
    ##     curr.max=-Inf
    ##     Gn.beginning.of.step = Gn
    ##     for(ii in 1:sum(!sapply(Tcurr, is.null))){
    ##         Gn.beginning.of.this.node = Gn
```

```
##         j = Tcurr[[ii]][1]
##         k = Tcurr[[ii]][2]
##         if(Ecurr[j,k]-Scurr[j,k]<=1) next
##         cusums = binSegInf:::getcusums(s = Scurr[j,k],
##                           e = Ecurr[j,k],
##                           y = y)

##         ## Characterize signs
##         signed.cusummat = (cusums$contrasts) * (cusums$signs)
##         G[(Gn+1):(Gn+nrow(signed.cusummat)),] = signed.cusummat
##         Gn = Gn+nrow(signed.cusummat)

##         ## Find cusum maximizer
##         Bcurr[j, k] = cusums$bmax
##         Zcurr[j, k] = cusums$signs[cusums$bmax.cusums]
##         breaking.cusum = cusums$cusum

##         ## Keep running maximum
##         if(curr.max <= breaking.cusum){
##             curr.which.max = c(j,k)
##             curr.max = breaking.cusum
##             curr.max.signed.row = signed.cusummat[cusums$bmax.cusums,]
##             curr.max.signed.rownum = Gn.beginning.of.this.node + cusums$bmax.cusums
##             curr.max.cusums = cusums$allcusums## temporary, for debugging
##         }
##     }

##     ## Record maximizer row and rownum
##     max.signed.row = curr.max.signed.row
##     max.signed.rownum = curr.max.signed.rownum

##     ## Characterize cusum-maximizer
##     this.step.rows=(Gn.beginning.of.step+1):Gn
##     this.step.rows = this.step.rows[this.step.rows!=max.signed.rownum]
##     comparison.cusummat = t(apply(G[this.step.rows,,drop=FALSE], 1, function(myrow){
##         curr.max.signed.row - myrow }))
##     G[(Gn+1):(Gn+nrow(comparison.cusummat)),] = comparison.cusummat
##     Gn = Gn+nrow(comparison.cusummat)

##     ## Check characterization (to be continued)


##     ## Record knot as CUSUM maximizer
##     zetas[mystep] = curr.max
##     jmax = curr.which.max[1]
##     kmax = curr.which.max[2]

##     ## Update terminal and active node set
##     which.duplicate = which(sapply(Tcurr, function(myjk){all.equal(myjk, c(jmax, kmax))==TRUE}))
##     Tcurr[[which.duplicate]] <- c(jmax + 1, 2*kmax - 1)
##     Tcurr[[mystep+1]] <- c(jmax + 1, 2*kmax)
##     Acurr[[mystep+1]] <- c(jmax,kmax)
```

```
    ##     ## Update Scurr and Ecurr for the /new/ nodes
    ##     Scurr[jmax+1,2*kmax-1] = Scurr[jmax,kmax]
    ##     Ecurr[jmax+1,2*kmax-1] = Bcurr[jmax,kmax]

    ##     Scurr[jmax+1,2*kmax] = Bcurr[jmax,kmax]+1
    ##     Ecurr[jmax+1,2*kmax] = Ecurr[jmax,kmax]

    ##     trim = binSegInf:::trim
    ##     ## Take snapshot
    ##     S[[mystep]] = trim(Scurr)
    ##     E[[mystep]] = trim(Ecurr)
    ##     A[[mystep]] = trim(Acurr)
    ##     Tt[[mystep]] = trim(Tcurr)
    ##     B[mystep] = Bcurr[jmax,kmax]
    ##     Z[mystep] = Zcurr[jmax,kmax]
    ##     jk[[mystep]] = c(jmax,kmax)

    ##     if(verbose) cat("From candidates", Scurr[jmax,kmax],  ":",Ecurr[jmax,kmax], " ")
    ##     if(verbose) cat("breakpoint", Bcurr[jmax,kmax], "was selected", fill=FALSE)
    ##     if(verbose) cat(" with threshold knot", round(zetas[mystep],3), " !", fill=TRUE)

    ##     ## Terminate if all terminal nodes are length 2 or smaller.
    ##     too.short = unlist(lapply(Tt[[mystep]], function(mypair){Ecurr[mypair[1],mypair[2]] - Scurr[m
    ##     if(all(too.short)){
    ##         if(verbose) cat("Ended early, at step", mystep, fill=TRUE)
    ##         break;
    ##     }
    ## }
    ## ## END of Main loop
    ## G = trim(G,"row")

## a = binSegInf:::binseg.by.size(y, numsteps, verbose=TRUE)
## print(a$B)
```